Pre-deliverables

Pre-deliverable 1

Learning Objectives

1. I hope to gain enough real-time embedded software experience in order to confidently enter a control-focused internship this coming summer where I could potentially be working with embedded systems like 3D printers for use in prototyping parts for aerospace companies.
2. I look to apply some of the control concepts I have learned in past and present control classes like 2.004 and 2.14. This control knowledge can help me program our flight systems for our final project, and has been used at various levels to control many different space flight systems.
3. I hope to be able to write robust and useful real-time code and understand how to efficiently implement test cases to validate my code. The ability to write robust and tested code benefits my final flight systems and any future embedded systems projects I may undertake.

Pre-deliverable 2

1. **What version of Python is installed on Athena?**

Python 2.7

2. **How do you define a constructor in Python?**

You define a constructor in Python starting with the following line of code:

```
def __init__(self):
```

In the parentheses following __init__, you enter the parameters that you would like the user to define upon creating the class. For example, in the class LoopNote that I made in a previous project, you have the following code:

```
class LoopNote(object):
    def __init__(self, pitch, start, length=60):
        self.pitch = pitch
        self.start = (start-1)
        self.length = length
```

The parameters pitch and start must be defined by the user upon creating a LoopNote object. The parameter length can be defined by the user, but if the user chooses not to define it then it defaults to 60, as shown in the parentheses. Following the initial line of the constructor, the definition must initialize all instance variables that the object needs upon start-up.

Sources:

Previous Pset 2 in 21M.385 taught by Eran Egozy

http://www.dummies.com/programming/python/how-to-create-a-constructor-in-python/

3. **How do you raise exceptions in Python?**

You raise an exception in Python by using the raise function followed by the exception you would like to raise. For example, you can raise a ValueError with the following line of code:

raise ValueError('z is 3')

This line is from the larger block of code that I wrote:

```
def test(z):
        try:
                if z == 3:
                        raise ValueError('z is 3')
                print("success")
        except Exception, arg:
                print'Error: ', arg
test(3)
```

In this code, the Value Error will be raised within the try portion of the try and except block if z is equal to 3. Otherwise the function will continue and print “success”.

If the raise line was outside of the if statement, then the print statement would not be reached.

Sources:

https://infohost.nmt.edu/tcc/help/pubs/python/web/raise-statement.html

https://wiki.python.org/moin/HandlingExceptions

4. **When using the Python unit testing framework, how can you check if a particular exception is raised during execution?**

You can check if a particular exception using assertRaises(exception). For example, as explained in the Python docs regarding unit test cases, the following line of code may check for SomeException:

```
with self.assertRaises(SomeException):
    do_something()
```

if SomeException is raised, then the code will execute the following line do_something(). This returns an error if another exception is raised, and fails if no exception is raised. A tuple of exceptions may also be passed if the need exists to catch one of a group of exceptions.

Source: https://docs.python.org/2/library/unittest.html

5. **How do you define a “main” function in Python?**

You define a main function using the following code:

def main():

    #more code to execute upon running main here

You run a main with the following line of code:

```
if __name__ == "__main__":
    main()
```

Source: http://stackoverflow.com/questions/22492162/understanding-the-main-method-of-python

Time spent on this question: 1 hour

Pre-deliverable 3

In python file

Pre-deliverable 4

Do not require unit tests:

1. getSpeed: It is a simple method that can easily be tested without a test case.
2. getRotVel: It is a simple method that can easily be tested without a test case.
3. getPosition: It is a simple method that can easily be tested without a test case.
4. getVelocity: It is a simple method that can easily be tested without a test case.

Requires unit tests:

1. Control constructor: In order to check that all boundaries are obeyed.
2. GroundVehicle constructor: In order to check that all boundaries are obeyed and that the velocity is calculated correctly.
3. setPosition: In order to check that all boundaries are obeyed.
4. setVelocity: In order to check that all boundaries are obeyed and that the correct action is taken when the input exceeds the boundaries.
5. controlVehicle: In order to check that all boundaries are obeyed and that the correct action is taken when the input exceeds the boundaries.
6. updateState: To ensure that the dynamics model does indeed produce the correct curved trajectory.

Input Equivalence Classes

Control(s, omega)

- s<5
- s>10
- 5 <= s <= 10
- Omega > -pi/4
- Omega < pi/4
- -pi/4 < omega < pi/4

GroundVehicle(pose, s, omega)

- 0<X position < 100, 0<Y position < 100, -pi<theta<pi
- X position >100, Y position >100, theta > pi
- X position < 0, Y position <100, theta < -pi
- For all conditions, s = 7 and omega = 0 because they are saved as a Control object and tested as such.

The rest is in the python file.